POCKET COMPUTER NEWSLETTER

© Copyright 1982 — Issue 13 March

THE SHARP PC-1500

An actual Sharp PC-1500 Pocket Computer arrived at *PCN* headquarters just a few hours before this issue was scheduled to go to press. This report, thus, represents a quickly compiled summary of our initial findings and reactions.

It Is Nice

This is definitely a second-generation PC! One of the first features one notices is the improved ease and speed with which program editing can be accomplished. A lot of the improvements are subtle but definitely noteworthy.

For instance, there are just two basic machine modes: RUN and PROgram. You only have to hit the mode key once to switch from the one to the other, instead of cycling through four modes as in the PC-1211. (There is a third mode, the RESERVE mode. However, since this is generally used far less often, this mode is achieved by pressing the SHIFT/MODE keys.)

Editing execution is virtually instantaneous. If you want, say, line 40 listed in a 40 line program, that line comes up in the blink of an eye. There is no three or four second delay to access the line. The cursor and line scroll operations are also much speedier. You can zip from beginning to end of a 40 line program in just a few seconds using the line scroll. Character scrolling across a line is also faster than on a PC-1211. This snappier editing operation is certain to be appreciated by programmers.

Perhaps the nicest operating/editing feature initially noticed is the implementation of "softkeys." There are six keys directly under the display that are user-programmable. They can act in a manner similar to the "defined" keys on a PC-1211, but with several significant advantages. These keys are "programmed" by placing the PC in the RESERVE mode. However, each key can actually be programmed for up to three different options. The desired option is selected by a Reserve Select key which is used to cycle to the desired option. The status of the Reserve Select key is denoted by indicators on the display which show the Roman numerals I, II or III. The following characteristics make these software keys particularly powerful.

First of all, they only require the pressing of a single key, instead of first having to press the SHIFT button.

Now, these keys can be used in the editing mode, for instance, to insert often used statements in a program under development. Or, they can be used to place a command on the display. However, through the use of a special terminating character (@), these keys can be set up to actually execute a command. For instance, setting up a reserve key to contain the directive:

RUN 100 @

results in your having the capability to execute a program starting at line 100 simply by pressing one key. (This same operation would have taken three keystrokes on a PC-1211: pressing the SHIFT key, the defined key and then the ENTER key.)

But wait, that is not all! With six softkeys, each of which can carry three functions or operations, it could be rather difficult to remember what had been assigned to all the keys. No problem. Whenever you assign capabilities to softkeys you can also create labels that appear directly above the keys on the display! To access those labels you just press the key marked RCL (recall)! It really brings a whole new level of operating comfort to the PC.

Here Comes Speedy

By PC standards, the Sharp PC-1500 is fast. Three benchmark tests conducted against the PC-1211 showed program execution to be 7 to 8 times faster!

A True Graphics Display

People are going to have a lot of fun with this display. Even in the text mode it has capabilities and advantages over the PC-1211. For instance, using the PAUSE statement (in a repetitive loop) simply causes the display to be updated. There is no more annoying blanking (unless you program that to occur). You can easily locate text anywhere along the display using the CURSOR directive.

Then, of course, there is the ability to address all of the individual points (7 by 156 dots) in the display matrix. Virtually any pattern desired can be obtained through the use of the GCURSOR and the GPRINT statements. The status of a dot can be obtained by using the POINT directive. A test routine using a loop to draw a line all the way across the 156 dots of the display executed in a little under 5 seconds. That amounts to about 30 pixels a second when the matrix is accessed on a point basis. Graphics can actually be drawn on a column basis, so in this mode some 200+ pixels-per-second can be manipulated. The impression of speed, in for instance, animations, can be enhanced by skipping columns or rows as an object is moved.

The display can be switched to lower case letters using the SML (small) key. And, with the graphics capability, you can actually design your own character set for special applications, if desired.

The Clock

The PC-1500 has a special function named TIME. You can set the time to the month, day, hour, minute and second. Once set, this built in clock is automatically maintained by the PC. You can then access the time as a function under program or keyboard control. This has obvious applications in programs such as appointment organizers and reminders. You can periodically check the time and then issue an audio signal when appropriate.

Music To Your Ears

The BEEP function is greatly expanded over that of the PC-1211. Not only can the number of beeps be programmed, but also the tone (256 frequencies) and the duration (from just a "pip" to more than a minute). Though the volume is discrete, it is easy to program special annunciator effects and those with the interest can make this little PC play music.

There Is Lots More

It is all described in two manuals that come with the unit. A 160+ page Instruction Manual with numerous examples, plus a 200+ page Applications Manual that contains about 50 complete programs.

The Initial Summary

It is nice.

DETERMINE TAB SETTINGS AUTOMATICALLY

If you have ever had to type charts or data tables, you know how frustrating it can be to arrange columns on a page neatly. Where should you set the tabs? How wide should the margins be? Usually it takes several trial runs before things fit properly. But, with your pocket computer and this program, you can literally take the guesswork out of designing tables.

This program allows columns to be the same or different sizes. The program also compensates for either elite type (102 characters across an 8-1/2 inch wide sheet of paper) or pica type (85 characters on a standard page). You can even vary the size of the paper if you want. Simple modifications will also enable the program to be used with font sizes other than those mentioned.

What You Need to Know

To use the program, you need to do just a little preliminary thinking. The program will want to know:

1. The number of columns across the sheet.
2. How many characters in each of the columns – so that a total in all of the columns can be obtained.
3. The horizontal size of the paper. (The program defaults to 8-1/2 inches if you do not specify a size.)

Here Is an Example

For this illustration there will be 6 columns across the paper. For the sake of simplicity all the columns will have 6 character positions. The paper width will be the standard 8-1/2 inches.

Place the PC in the DEF mode. Press SHIFT/C. This entry point initializes all variables in the program. Respond as illustrated here:

Computer Shows	What You Do
RETURN TO INITIALIZE PGM	Press ENTER.
PICA OR ELITE (P/E)?	Enter E and press ENTER.
TABULATION DESIGNER	Program is now in the regular mode. It is only necessary to enter the initialize mode when you change type sizes. Press the ENTER key.
USING ELITE TYPE.	Press ENTER.
HORIZONTAL INCHES?	Press ENTER since computer will default to the 8-1/2 inch size paper being used for this example.
USING 8.5 INCHES.	Press ENTER.
PREF. LEFT MARGIN.	How wide (number of characters) do you want the margins to be? The program will warn you if you make the margins too wide. Input 10 and press ENTER here.
HOW MANY COLUMNS?	This example has six. Input 6 and press the ENTER key.
ALL THE SAME SIZE?	Press Y and the ENTER button for this case.
COLUMN WIDTH (SPACES)?	Enter the number 6 and press the ENTER key.
MAX. SPCS./COL.=9.2	The program indicates that there will be about 9 spaces between each of the six columns.

If the Spacing Is Not an Integer Value

Few typewriters can perform fractional spacing. If spacing between

Program *Automatic Tab Locator*

```
1  "C" PRINT "RETURN TO INITIALIZE PGM":CLEAR
2  Q$="X":INPUT "PICA OR ELITE (P/E)? ";Q$
3  IF Q$="E" LET S=12:Q$="ELITE"
4  IF Q$="P" LET S=10:Q$="PICA"
5  IF Q$="X" PRINT "ERROR!":GOTO 2
6  " " PRINT "TABULATION COMPUTER"
7  PRINT "USING ";Q$;" TYPE."
8  H=8.5:INPUT "HORIZONTAL INCHES? ";H
9  PRINT "USING ";H;" INCHES."
10 H=S*H
11 INPUT "PREF. LEFT MARGIN? ";M
12 INPUT "HOW MANY COLUMNS? ";T
13 INPUT "ALL SAME SIZE (Y/N)? ";R$
14 IF R$="N" GOTO 18
15 INPUT "COLUMN WIDTH (SPACES)? ";W
16 FOR I=1 TO T:A(I+26)=W:NEXT I
17 GOTO 23
18 FOR I=1 TO T:A=I+26
19 PAUSE "COLUMN #";I
20 INPUT A(A):IF A(A)>=1 GOTO 22
21 GOTO 19
22 NEXT I
23 X=0
24 FOR I=27 TO (T+26)
25 X=X+A(I)
26 NEXT I
27 W=X+(T-1)
28 IF W < H GOTO 30
29 PRINT "MIN. SPACES REQD: ";W:GOTO 6
30 D=H-(2*M)
31 IF D > X+(T-1) GOTO 38
32 PRINT "MARGIN TOO LARGE. . ."
33 V=(D-(X+(T-1))/2:PRINT V;"=MAX. AVAIL. MARGIN"
34 N=0:INPUT "PLEASE RE-ENTER MARGIN: ";N
35 IF N=0 GOTO 34
36 M=N
37 GOTO 30
38 U=(D-X)/(T-1)
39 PRINT "MAX SPCS/COL=";U
40 R$="X":INPUT "RE-DEFINE MARGIN (Y/N)? ";R$
41 IF R$="X" THEN GOTO 40
42 IF R$="N" THEN GOTO 46
43 PRINT "OLD MARGIN=";M
44 N=0:INPUT "NEW MARGIN? ";N
45 M=N:GOTO 27
46 PRINT "LEFT MARGIN=";M
47 L=M
48 FOR I=2 TO T
49 L=L+U+A(25+I)
50 USING "####":PRINT "COLUMN #";I;" =";L
51 NEXT I
52 PRINT "---END RUN":GOTO 6
```

columns is not an integer value, as in this example, use the rounded value as the best choice. Or, you can adjust the between column spacing by re-defining the margins. For instance, the example problem could be continued by pressing the ENTER key again:

Computer Shows	What You Do
RE-DEFINE MARGIN (Y/N)?	Input Y and press ENTER.
OLD MARGIN=10.	Press ENTER.
NEW MARGIN?	Input 8 and press ENTER.
MAX. SPCS./COL.=10	Good! Have the even spacing that was desired. Press ENTER again.
RE-DEFINE MARGIN (Y/N)?	Input N so can exit with solutions. Press ENTER.
LEFT MARGIN=8	Program reminds you where to start. Press ENTER.
COLUMN # 2 = 24	Press ENTER.
COLUMN # 3 = 40	Press ENTER.
COLUMN # 4 = 56	Press ENTER.
COLUMN # 5 = 72	Press ENTER.
COLUMN # 6 = 88	Press ENTER.
---END RUN	Press ENTER to return to the start of the program, if desired.

It Can Be Handy

If you are involved with a lot of statistical typing, accounting work or if you design report formats for computerized jobs, this utility program can save you a lot of time and trouble.

Submitted by: *Ken Slaughter, 2916 Bangor Avenue, Highland, CA 92456.*

AMORTIZATION PROGRAM

This program, submitted by *Emerich Auersbacher,* produces a complete loan amortization schedule. All you do is input the amount of the loan, the number of monthly payments and the annual interest rate. The pocket computer will then give you the monthly payment needed to repay the loan. It then gives the amount of interest due each month, the amount by which the principal is reduced each month, and the principal balance at the end of each period.

Planning on buying an appliance, car or house in the near future? Take your PC with this program in it along with you!

Program *Amortization*

```
5  BEEP 1:PAUSE "  AMORTIZATION SCHEDULE"
10 CLEAR:INPUT "LOAN AMOUNT $:";V,
   "PERIODS:";N,"INTEREST RATE (%):";Y
15 Y=Y/100,Y=Y/12
20 P=V*(Y/(1-(1+Y)^(-N)))
25 BEEP 1
30 PRINT "PERIODIC PMT=  $";USING "#####.## ";P
35 L=V
40 FOR Z=1 TO N STEP 1:USING
45 K=(1+Y)^(-Z),B=1/K*(P*(K-1)/Y+V):IF B < 1E-3 LET B=0
48 BEEP 1:PAUSE "PERIOD  ";Z
50 I=B-L+P,L=B,R=P-I
60 USING "#####.## "
65 PRINT "I=";I;"    P=";R:USING "########.## "
68 PRINT "BAL=$";B
70 NEXT Z
80 END
```

FAST-FOURIER-TRANSFORM

The Fourier transform is becoming increasingly popular as a mathematical tool as programmable calculators, and now pocket computers, have gained in capability. The Fourier transform becomes more useful as the number of data points increases. However, the amount of computation increases as the square of the data. Using the "fast" implementation reduces the number of computations required and greatly speeds the problem-solving process.

The compact fast-Fourier-transform program provided here leaves 20 steps of memory free even when 64 complex points are specified as data. (Thus, there is room to modify the program slightly if you desire a different input or output format, etc.)

Only 11 registers are used to store intermediate values, serve as loop counters and other program parameters. Execution speed is quite fast: 4.5 minutes for a 32 complex-point transform, 10.5 minutes for a full 64 complex-point transform.

Operation

Enter the RUN command. Respond to the prompt for "N" with the number of data points to be entered (maximum of 64). Respond to the prompt for "D" by inputting 1 for the forward transform (time to frequency) or –1 for the inverse (frequency to time).

The program is designed to use the printer unit. The data classification (FREQ or TIME) is printed as appropriate and the input of data requested. All points, including zero, must be entered. Data is printed for verification.

When all data points have been entered a bit-reversal routine is executed. Then the transform is performed. The RADIAN statement in line 4 of the program separates the bit reversal routine from the transform instructions.

Finally, the results are outputted to the printer.

Remember that the number of data points specified for a transform should always be a power of two, thus you should specify and enter either 2, 4, 6, 8, 16, 32 or 64 points. The more points used, the more accurate and meaningful the results.

For comparison purposes, you might want to review the FFT program designed for a TI-59 (Texas Instrument's) calculator that appeared on page 233 of the May 10, 1980 issue of *Electronic Design.* The TI-59 version required 13 minutes to solve a 32-point transform. The program presented here performs the same transform in one-third the time.

This program submitted by: *A. D. Chamier, 6 Cissbury Hill, Crawley, Sussex, England RH11 8TJ.*

Program *Fast-Fourier-Transform*

```
1  INPUT "N=";A,"D=";D:B=1:C=1:GOSUB 7:I=1:FOR K=1
   TO A-2:J=A
2  IF I > K LET E=11+K:F=11+I:G=A(E):A(E)=A(F):A(F)=G:
   H=A(E+A):A(E+A)=A(F+A):A(F+A)=H
3  J=J/2:IF J < I LET I=I-J:GOTO 3
4  I=I+J:NEXT K:RADIAN:FOR K=1 TO A/2:E=A-2K:FOR
   J=12 TO 11+K:B=(J-12)Dπ/K:C=COS B:B=SIN B
5  FOR I=J TO J+E STEP 2K:F=I+K:G=CA(F)-BA(F+A):H=
   CA(F+A)+BA(F):A(F)=A(I)-G
6  A(F+A)=A(I+A)-H:A(I)=A(I)+G:A(I+A)=A(I+A)+H:NEXT I
   :NEXT J:K=2K-1:NEXT K:D=-D:B=1:IF -D LET B=A
7  IF D PRINT "TIME
8  IF -D PRINT "FREQ
9  PRINT "N    REAL    IMAG":FOR K=12 TO 11+A:J=
   K+A:IF C INPUT "REAL=";A(K),"IMAG=";A(J)
10 G=A(K)/B:H=A(J)/B:PRINT USING "###.";K-11;USING
   "##.### ";G;H:NEXT K:IF C RETURN
```

USE OF MEMORY IN THE RADIO SHACK TRS-80 PC

The major portion of the random access memory in the pocket computer resides on three chips, each of which contains 512 bytes (capable of storing 1024 hexadecimal digits). The first 48 bytes are used for the RESERVE mode memory. The next 1424 are the regular memory used for storage of a program or data. The next 32 bytes store the variables W, X, Y and Z. The last 32 bytes are used to store information and addresses for the four levels of subroutines and nested FOR loops that the PC is capable of handling. (*PCN* Issue 06, page 4, contains the hexadecimal addresses of these memory locations.)

The variables A through V, the input buffer, and display buffer are stored on other smaller chips.

Data is stored in binary-coded-decimal (BCD) format. Sixteen hexadecimal digits are used to represent variable values. Two of these digits are used to indicate the signs of the exponent and mantissa. Two are used to store the value of the exponent. Ten are used to store the mantissa value. Two are left over.

String variables are stored in the same locations as numerical variables. One pair of digits identifies the variable as a string. The next seven pairs represent the seven characters making up the string. (*PCN* Issue 07, page 4, provides a list of the character codes used in the PC.)

Information in the RESERVE memory is stored in the following format: One byte marks the beginning of a segment. The rest of the segment is stored as character codes.

There are 256 different binary patterns that can be represented in an 8-bit byte. The PC uses about 110 of the possible combinations as tokens for the various program words and characters. Several other codes are used to mark the beginning and end of program lines.

The remaining possible codes, over 100, do not seem to be of much practical use. Many are displayed as duplicates of other characters. Some are undisplayable and appear to cause "crashes." An interesting exception is made by the codes AE and AF. These are displayed as STEPS and MEMORIES respectively. The unused codes print as question marks when fed to the printer, except for the codes AE and AF. These are spelled out as PI and SQR.

When the unused codes are placed in program memory and the program is executed, many of them simply produce an error. A few appear to act identically to some of the functions. For instance, the code 87 calculates arcsine. Others perform only part of a function routine. The code 80 seems to act like COS but does not always provide the correct sign. The codes 6B and 6F are equivalent to specifying A(27) and A(31). They might be of some conceivable use. However, the task of placing these codes into a program line is so complex and time-consuming, that it simply isn't worth the effort as a practical matter.

This information provided by: *Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158.*

EXPLORING THE RADIO SHACK AND SHARP PCs

An easy way to begin exploring what takes place inside these PCs is to use the fact that when program lines are deleted, the boundary between program storage and data is relocated, but the previously stored information is not erased! Thus, program lines may be deleted and the previously stored hexadecimal digits recalled as data or strings.

For instance, after clearing out memory with the directive NEW, try entering the program line:

25 N=LMQ

Then, delete line 25. (Just key in 25 and ENTER with the computer in the PRO mode.)

The program line just deleted will still be stored as the hexadecimal pairs:

E0 25 5E 34 5C 5D 61 00

(*PCN* Issue 05, page 3, describes how pairs of hexadecimal digits are stored as bytes.)

Next, with the PC in the RUN mode, type A(204) and press the ENTER key. This results in the unit displaying:

¥5.%543$552

This display may be explained as follows:

A variable storage location holds 16 hexadecimal digits. These are read from memory in the direction *opposite* to that in which program bytes are read. The first three digits specify the sign and magnitude of the exponent. The next eleven represent the sign and digits in the mantissa. The final two digits, serving as the 11th and 12th digits of the mantissa, are not displayed. Note that when displayed in this manner, the hexadecimal digits 0 through 9 are represented directly, however the hexadecimal symbols A through F are displayed as the characters . (period), E (exponent indicator), %, ¥, $, and π.

When the hexadecimal digits are transcribed to appear in reverse order, the result is:

00 16 D5 C5 43 E5 52 0E

This can be interpreted in the following manner: The first zero indicates a positive exponent value. The next two digits, 0 and 1, provide an exponent value of 1. This explains the decimal point location in the display of A(204) above. The fourth digit, 6, indicates the mantissa is positive in value. The next ten digits represent the mantissa value.

Those who try some experimenting will find that if the first digit is non-zero, the exponent is negative in sign. Negative values are represented in complementary form. If the fourth digit is 8 or greater, then the mantissa is negative.

As an example of storage of a string, enter the program line:

581 ADF+QO

Note that the last character is the letter of the alphabet, not the digit zero. Now delete this line. Displaying A$(204) will show: U%C5#$N. When decoded to hexadecimal pairs (see *PCN* Issue 07, page 4 for a conversion table) this yields:

E5 81 51 54 35 61 56 5F

Reversing this sequence provides:

F5 65 16 53 45 15 18 5E

The first byte here, starting with the hexadecimal symbol F, indicates that the variable is storing a string. The remaining seven bytes represent the characters in the string.

If you want to explore even further, try using the computer in a mode in which program memory overlaps data memory and the loop/subroutine registers. (Additional information regarding this technique is in *PCN* Issue 04, page 2, with a correction to the method discussed in Issue 07, page 6.

[Norlin Rober, 407 North 1st Avenue, Marshalltown, IA 50158, developed this method of exploring the PC. His pioneering work in this area is certainly appreciated by PC enthusiasts. – N.W.]

A MESSAGE FOR AUTHORS

PCN invites the submission of quality programs for pocket computers as well as practical application and tutorial articles of specific interest to users of pocket computers.

PCN purchases all rights to accepted materials at rates ranging from $50 to $150 per finished page as it appears in *PCN*. The payment rate is highly dependent upon the editorial and technical quality, timeliness and general suitability of the material for our readers.

Please double-space all manuscript material. Include a clean printer listing and a tape cassette copy of substantial listings. Be sure and include a self-addressed, stamped envelope if you desire the return of material that has been submitted for our consideration.

We keep all materials, including cassette tapes, and provide an Author's Agreement when a submission meets our requirements.

Please direct all submissions for editorial review to:

The Editor
POCKET COMPUTER NEWSLETTER
35 Old State Road
Oxford, CT 06483

POCKET COMPUTER INTERFACE SIGNALS

Ron Miller, 11918 S.E. Holgate Boulevard, Portland, OR 97266, has been doing some detective work on the signals used to interface with the Radio Shack TRS-80 PC. Here is what he has discovered to date. (The pin numbering convention is the same as that used previously in *PCN.* It is reviewed in an accompanying diagram.)

Pin 1: Serial data output to the printer. Data is transmitted at a rate of 500 baud. Each character is presented as two 4-bit "nibbles." A nibble is preceeded by a start bit. There are 4 stop bits between nibbles. Characters are separated by 5 stop bits. The first nibble in a character contains the four most significant bits. The second nibble contains the four least significant bits. Within a nibble, the least significant bit is sent first, the most significant is last. The character codes are the same as those previously listed by *Norlin Rober.* (See *PCN* issue 5, page 3.)However, line numbers are not preceded by the codes E0 — E9. Instead, the code 8E heads a string when a PRINT command is executed or 8D when a LIST command is in progress. Data strings are terminated by the code 00.

Pin 2: Indicates when the printer data buffer is full. It is sent from the printer and tells the PC to stop transmitting. It is normally low. It is brought high when the PC is to inhibit transmission. It appears as though the printer has an input buffer of about 80 characters.

Pin 3: When high, print statements are routed to the printer. This signal is apparently only sampled by the PC when the POWER ON key is pressed twice.

Pin 4: This signal notifies the printer that the PC has data to send. It is normally high. It goes low about 3.35 milliseconds before the first data appears on pin 1. It is kept low until about 8 milliseconds after the last character in a line has been transmitted.

Pin 5: This is the Vcc positive supply pin.

Pin 6: Relay control signal for controlling a tape recorder. Normally in a high state, the signal goes low whenever a tape read or write operation is executed. (See *PCN* issue 4, page 1.)

Pin 7: Audio input from a tape cassette to the PC. Format is the same as described for pin 8.

Pin 8: Output from the PC to a tape cassette interface. The signal is a square wave with a frequency of approximately 4000 Hertz representing a logic high and a frequency of 2000 Hertz for a logic low. The baud rate is 500.

Pin 9: This is the common (0 volt) reference.

Note: Apparently the printer recognizes statement tokens in the same manner as the PC display. Only the token code is transmitted and tokenized words are never broken up when printed.

For a practical application of the information described here, turn to the next page!

Figure *Pin Numbering Convention*

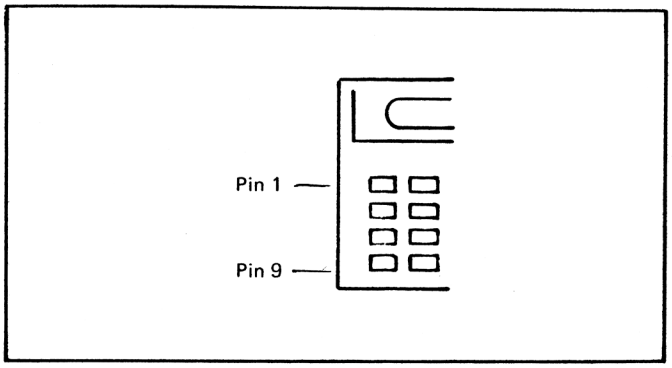


Figure *Printer "Handshaking" Logic Timing Diagram*

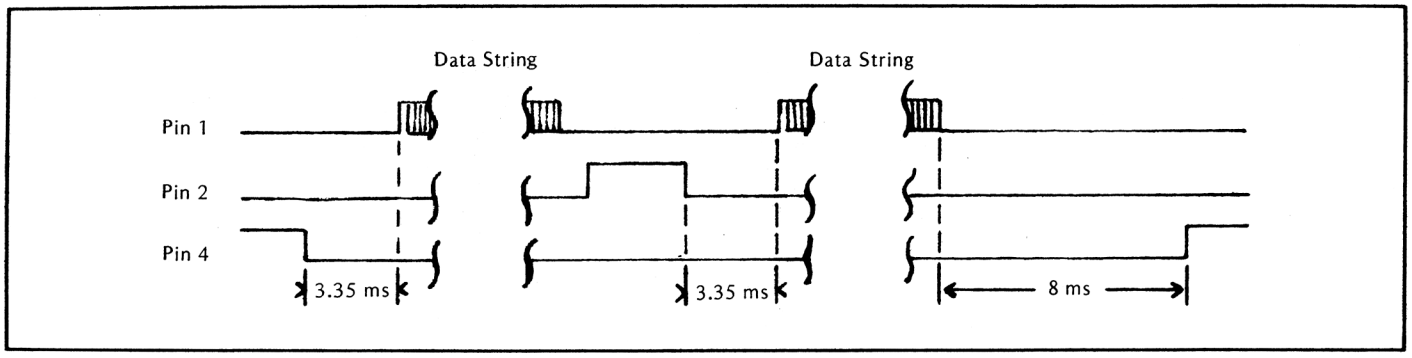


Figure *Format Used to Transmit Data to the Printer*

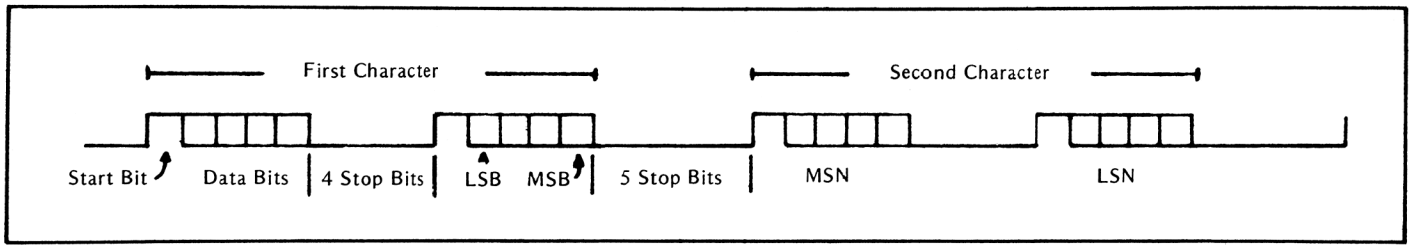
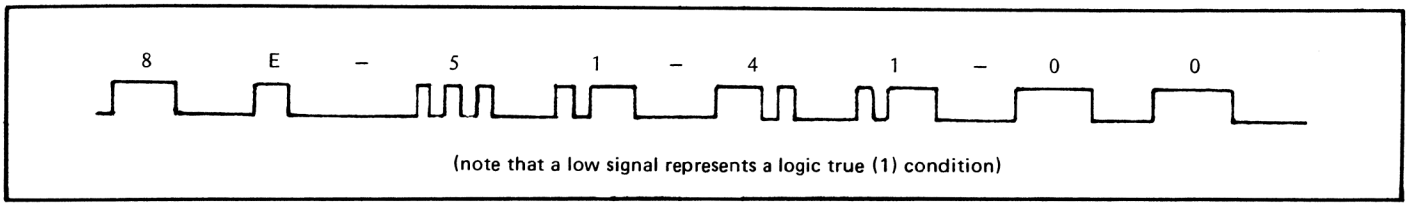


Figure *Waveform Transmitted to the Printer by Execution of the Statement:* PRINT "A1"



X-Y PLOTTER INTERFACE FOR THE SHARP PC-1211

If you have access to an analog X-Y plotter, you may be interested in building this interface. It allows you to output digital data from a Sharp PC-1211 or Radio Shack TRS-80 PC to the plotter.

The PC-1211 outputs data to the CE-122 printer in BCD (binary coded decimal) format using its own hexadecimal character set. The circuit shown in the accompanying schematic converts this digital data to an analog signal.

The circuit illustrated uses series 7400 low power TTL (transistor-transistor-logic) and two digital-to-analog converters. A separate low frequency transistor is used to isolate the pocket computer and thus avoid any possible damage to the unit.

The circuitry consumes about 330 milliamperes from its own 5 volt power supply. Much of the power is consumed by the 74148 integrated circuits. These devices could be eliminated if the data is converted to octal notation using software before it is transmitted from the PC.

To minimize the parts count, the X and Y data values are limited to the integer range 0 through 99. Because of the manner in which data is outputted, the data must be outputted in the format: 1XXYY. This is accomplished by multiplying the X value by 100, adding this to the Y value, and then adding 10000. The latter step prevents a leading zero in the X value from being blanked. These operations can be combined into a single statement:

```
PRINT E4+E2X+Y
```

This effectively formats the value to be outputted in the right hand side of the display complete with a trailing decimal point. This value is then outputted on the printer data pin when the PC is fooled into thinking it is connected to a printer.

When the PC is set to output data to a printer, it proceeds as follows: Each byte of data is transmitted as two 4-bit "nibbles." The transmission is preceded by the value "8E" that signifies that the communication is for the printer. Then the 24 positions of the LCD are sent (even though the display appears blank) proceeding from left to right. The transmission is terminated with the hexadecimal byte 00.

Each 4-bit nibble is preceeded by a "start" bit. Thus, the method is that of a simple asynchronous transmitter. The accompanying logic timing diagram clarifies the method.

The X-Y plotter interface uses a simply asynchronous receiver to clock data into shift registers. Since the first nibble of a decimal digit is always 4, these nibbles are blanked out to save hardware. At the conclusion of each data stream (which takes 0.85 seconds) the relevant nibbles representing the X and Y data are latched into the 74LS-273 ICs. This data, now in parallel format, is then decoded to binary format and fed to the DACs (digital-to-analog converters) ready for outputting to the plotter.

While the data receiver may appear complex because it uses three dual monostables, this configuration actually provided the lowest IC count. It may be necessary to adjust the timing resistor values slightly to match the tolerances of the capacitors used as well as to compensate for variations between PC units, since the PC-1211 does not utilize a crystal clock. Only the monostable oscillators marked D and E in the schematic are really critical. The functions of the monostables are outlined here:

The "word" monostable (W) remains triggered throughout the data stream. When the stream ends, the new data is latched.

The "initial" monostable (I) blanks the initial nibble of each byte so that only the relevant digit-indicating nibbles are captured.

Monostable (A) blanks the first nibble of each byte.

Monostable (B) resets the bit counter and times out the "start" bit that precedes each nibble.

Monostables (D) and (E) serve as an oscillator to clock in the four bits of each nibble to the shift registers.

Counter (C) stops nibble data entry after 4 bits are received.

In practice, small slew movements will be drawn as straight lines between the two data points. Large slews may result in curved lines. The capacitor filter provided on the output of the DAC delays the onset of this effect. The capacitor value shown provides an RC time constant of 220 milliseconds for the DAC used and gave good results. (Remember, it takes at least 0.85 seconds between new data values due to the manner in which the PC serially transmits data.)

The BASIC program that provides the data to be plotted must scale the data points to be integer values in the range 0 to 99. The print statement presented previously can then be used to output the scaled data. The use of exponent notation and implied multiplication in the PRINT statement saves four bytes of program space.

Direct PRINT statements can also be used to draw figures or characters. For instance, graph axes might be displayed using the series:

```
PRINT E4:PRINT 19900:PRINT 19999:PRINT 10099:PRINT E4
```

Pen lifting and lowering must be performed manually. However, this can be prompted in the PC's display using a dummy input statement such as:

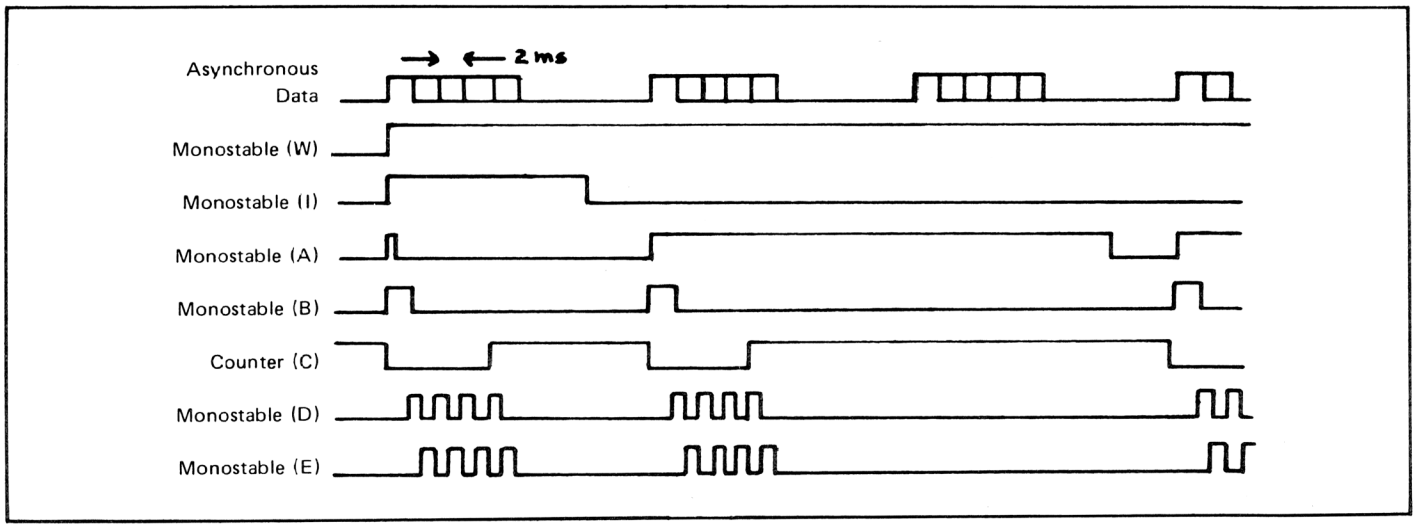
```
INPUT "LOWER PEN";K
```

A possible, option for those that want to dabble some more with the hardware, would be to use the decimal point position as a pen status flag. The USING format could be used to allow any digit to be placed as the last character (in place of the decimal point). The value of this digit could be used to raise or lower the pen.

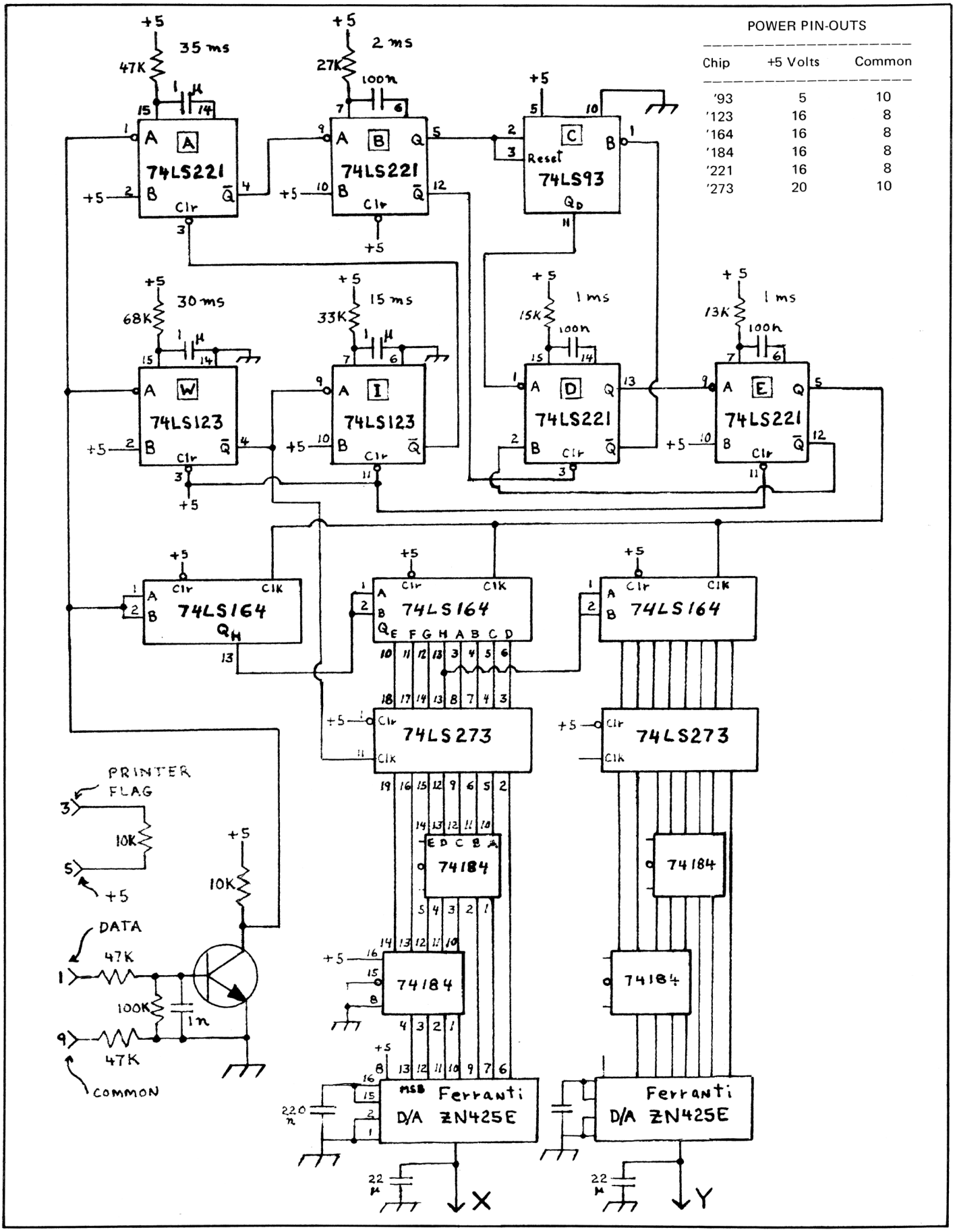
Other options include simply eliminating the binary converters and DACs and using the remaining circuitry as a simple parallel output device to control "whatever."

Thanks for this design go to: *A. D. Chamier, 6 Cissbury Hill, Crawley, Sussex, England RH11 8TJ.*

Figure *X-Y Plotter Interface Logic Timing Diagram*



Schematic *X-Y Plotter Interface*



POWER PIN-OUTS

Chip	+5 Volts	Common
'93	5	10
'123	16	8
'164	16	8
'184	16	8
'221	16	8
'273	20	10

PROGRAMMING TIPS AND TRICKS

Have you ever had a problem with overflow on your pocket computer? If you deal extensively in the fields of science, engineering or mathematics, where large numbers are frequently used, chances are good that you have left problems unsolved because your computer overflowed. Although final answers are generally likely to be within the range of a PC (less than E100), intermediate results in many cases can cause the computer to overflow.

Want to eliminate those overflow aborts? Sure you do. Here is a way to get rid of most such situations: The trick is to use logarithms in all of your calculations where overflow is likely to occur. Then use the output routine presented here to display the results.

Suppose you were dealing with factorials. A conventional routine to solve for factorial values might appear as follows:

```
10 F=1:INPUT "N?";N
20 FOR Z=1 TO N
30 F=FZ
40 NEXT Z
50 PRINT "N!=";F
60 END
```

Unfortunately, this standard technique overflows when N exceeds 69. If you want to compute 350!, you are out of luck.

However, if the problem is set up using logarithms, a routine for processing factorials could appear as shown here:

```
10 F=LOG 1:INPUT "N?";N
20 FOR Z=1 TO N
30 F=F+LOG Z
40 NEXT Z
50 E=INT F:M=10^(F-INT F)
60 USING:PRINT "N!=";M;"E";E
70 END
```

Now the factorial products become sums. Lines 50 and 60 convert and format the results for output. (The USING statement is recommended, when lines 50 and 60 are used as a subroutine, to cancel any previous format.) Check it out for yourself. Inputting 350! to the above routine yields 1.235873158 E 740.

One small note of caution: The technique may result in the loss of one or more significant digits in the mantissa, depending on the size of the exponent. Generally, of course, this result is still much better than not being able to get any answer at all!

Speaking of Factorials

The following asymptotic series approximation can be used for the Gamma function:

$\Gamma(N)=e^{-N}N^{N}\sqrt{(2\pi/N)}[1+1/12N+1/288N^{2}-139/51840N^{3}-\ldots+\ldots]$

For positive integers:

$N!=\Gamma(N+1)$

Satisfactory results for all N greater than or equal to 1 can be obtained using the first 3 terms of the above series. Now a factorial program can be written in the form:

```
10 INPUT "N?";N
20 N=N+1,F=INT(EXP-N*N^N+√(2π/N)*(1+1/12N+1/288NN))
30 PRINT "N!=";F
40 END
```

What this method does is eliminate the loop used in the previous routines. Now all values of N take approximately the same amount of time to calculate. By the way, statement 20 can be used as a general factorial subroutine. Or, you can put it into REServe memory, assign it to the N key and have your own N! button.

Of course, if you want to determine factorial values for N greater than 69, the routine can be re-written to use the log method described earlier:

```
10 INPUT "N?";N
20 N=N+1:F=-N*LOG EXP 1+N*LOG N+LOG(2π/N)/2+
   LOG(1+1/12N+1/288NN)
30 E=INT F:M=10^(F-INT F)
40 USING:PRINT "N!=";M;"E";E
50 END
```

This routine can save quite a bit of time, yielding answers in seconds instead of minutes. Furthermore, the limit on FOR–NEXT loops is 999, while this routine has no such limit.

Engineering Notation

Many people find it convenient to be able to have answers expressed in engineering (not scientific) notation. In this type of notation, the mantissa is normalized so that it is in the range of 1 to 1000 and the exponent is always displayed as multiples of 3. This way, the units of measure are more readily apparent. Thus E-6 can be related directly to micro (as in microfarads) or E-9 to nano(seconds), E03 to kilo, E06 to mega(cycles) and so forth.

A routine to format a PC display to engineering notation is shown here:

```
900 Z=LOG ABS X:USING
910 E=INT Z:M=10^(Z-INT Z)
920 IF E/3 < > INT(E/3) LET E=E-1:M=10*M:GOTO 920
930 M=M*SGN X:PRINT M;"E";E
```

To check this out to your own satisfaction, add a statement to input values (X) to the start of the above routine, add an END statement to conclude the routine and try a few values. You should see that:

3.25E7 becomes 32.5E6

1.25E-7 becomes 125E-9

Thanks for these choice tidbits go to: *Emerich Auersbacher.*

FROM THE WATCH POCKET

As you can see from the front page of this issue, the Sharp PC-1500 is now actually available in the United States. I have also ascertained that it is now also available in Canada. The word is that the CE-150 combination Printer/Cassette Interface will be available here, as will the 4K memory module, within just a few days.

With just a few hours to look the unit over, I am initially impressed. It looks as though Sharp has done their homework on this model and put in a lot of the features that PC-1211 users have indicated they wanted in the next generation of pocket computers.

One little word of caution to beginners who get this unit. It looks as though the manuals that come with the PC-1500 may have been a little rushed. Some of the example routines/programs contain the types of errors that indicate they were not thoroughly tested. So, don't be surprised if a few examples don't work quite as stated. There is a lot of material in these manuals, however, and most of it is undoubtably correct.

On To Other Subjects

H. Walker wants PC users to know that Radio Shack has advised him not to try using the CTR 80 with the Radio Shack TRS-80 Pocket Computer when the recorder is powered by an alternating current source. Could be that hum or line noise interferes with the proper loading of programs. You might want to keep this advice in mind if your are having program recording or loading problems with any tape recorder that is operated by house current. Try using a battery operated tape recorder. If this gives you improved performance, stick with it!

After praising **80 Microcomputing** for being one of the few magazines to carry articles and programs on PCs, I was unable to find a single such article in the February issue!

But, *Bob McElwain* is still plugging away in the February issue of **Interface Age**. The program presented there can, among other things, calculate how much interest you paid on a mortgage over a specified time span.

When is Radio Shack going to have their PC-2 (equivalent of the Sharp PC-1500) available? Scuttlebutt I have heard puts it at sometime around May. If that turns out to be the case, Sharp could do a nice volume before then. — *Nat Wadsworth, Editor*

35 Old State Rd, Oxford, CT 06483